Middle East Technical University
Department of Computer Engineering

CENG491
Computer Engineering Design I
Design Report

ÖZGÜR YAZILIM

Özgür Özgür
Fırat Erdoğan
Onur Demircan
Abdulkerim Mızrak

1. PROJECT DEFINITION

Treasure Hunt is a massively multiplayer online game (MMOG) with 3D graphics. It is designed to support hundereds of online players to interact and play together in a virtual environment.

The game has a nice scenario. Players are restricted in a multi-storey building with lots of rooms connected by several paths. Players are given the chance of choosing one of the pre-defined characters. In the game, characters have calorie values increase with the food they eat. On the other hand as time passes, movement and metabolism result in the decrease of calorie values.There are some predefined calorie limits which determine the characters movement capability. According to calorie value character can walk or run or crawl.There are some food objects that characters can compensate the energy loss by eating. Characters have special characteristics, for example, some characters are vegeterian. Vegeterian characters can not eat meat. Moreover, some characters are allergic to some kind of food. Players dedicate themselves to find the treasure. Players approches the TREASURE, wandering inside the building and passing steps by finding step related treasure objects. Finally the one most skillful, fast and wise hunter will reach the TREASURE.

2. DESIGN GOALS

Treasure Hunt is going to be a massively multiplayer online game which supports 3D Graphics Rendering, Multimedia (sound), and Game AI.

In this report, our main goal is to show the design made by our group in order to document our project properly. This report is going to describe a detailed functionality of the system, main components of the system and their interaction with each other, graphical user interfaces, and a gantt chart.

In general, we will not only try to implement a game with successfully played in any way, but also try to satisfy the following software design goals as well.

- **Object-Oriented –** Object oriented design results in self-contained modules that are easy to manage and maintain. The modular design of Treasure Hunt makes it easy to implement, test, maintain, and extend.

- **Extensibility –** The system evolution for games such as Treasure Hunt is endless, so the design should allow for the implementation of future requirements with minimal changes to the current design. We are planning to build a system with ‘plug-in’ property. So that, we will add new objects easily to the system.

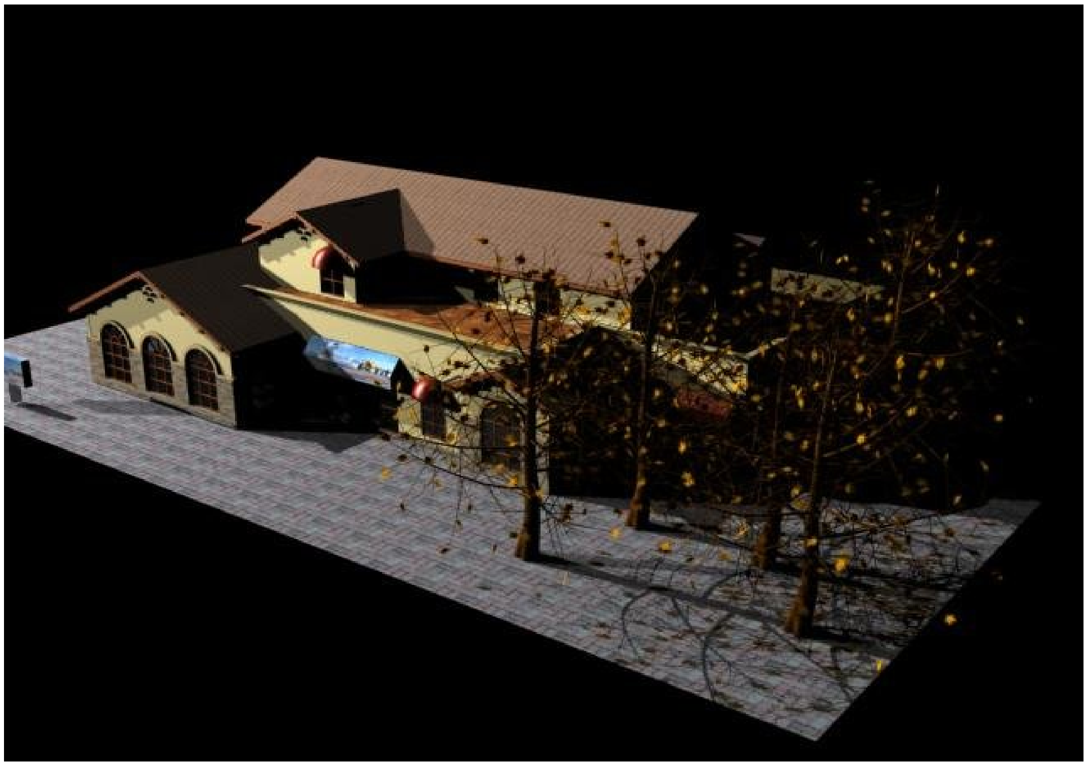
- **Ease of use –** It is important that the application has a clean and easy to use interface. The interface should be intuitive and look very familiar to both new and experienced gamers.

- **Performance** Treasure Hunt is designed to minimize the resources occupied and to maximize the output in terms or frames per second. The game is designed to be responsive, robust, and error-free in any system that meets the minimum system requirements specified in the non-functional requirements.

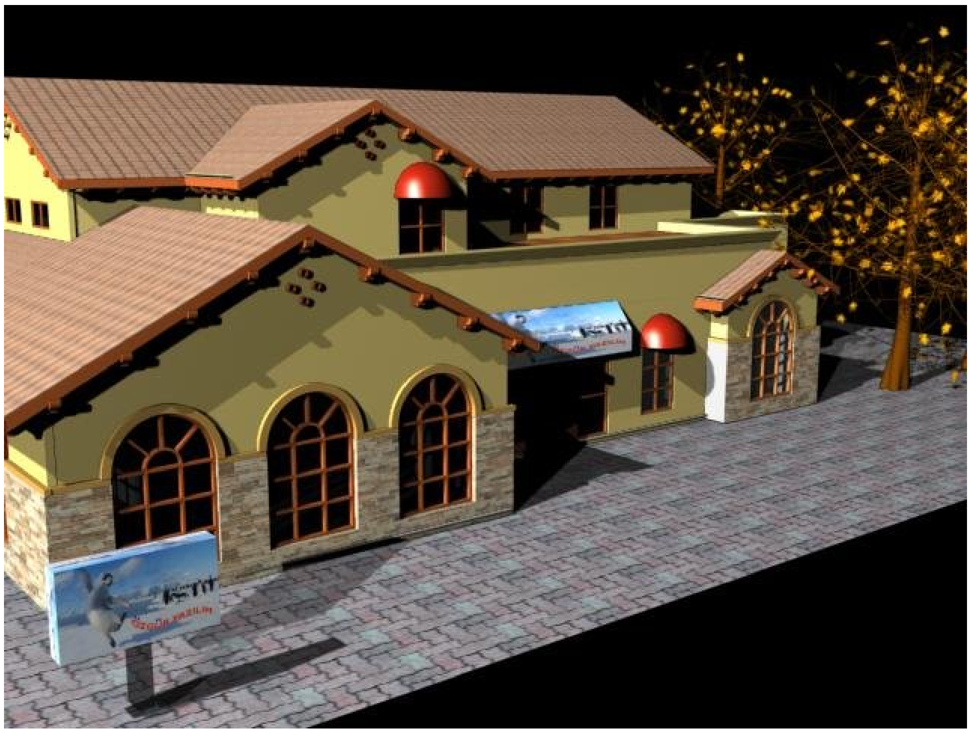
3. MODULES

3.1 Graphical User Interface

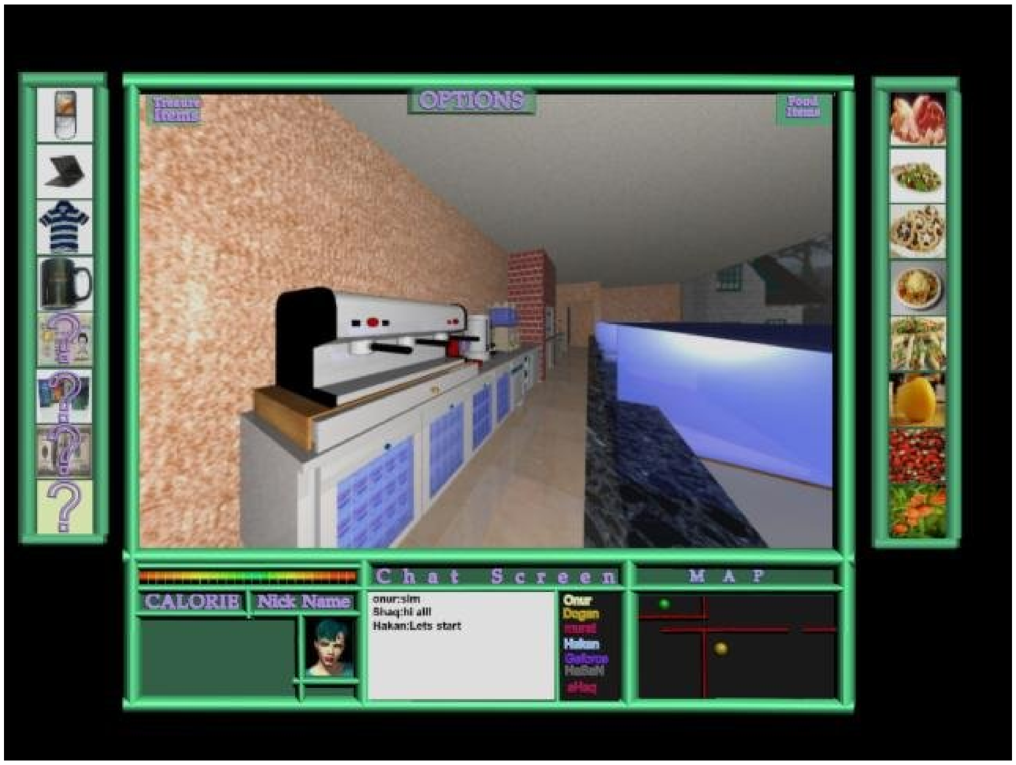
Interface 1.



Interface 2.

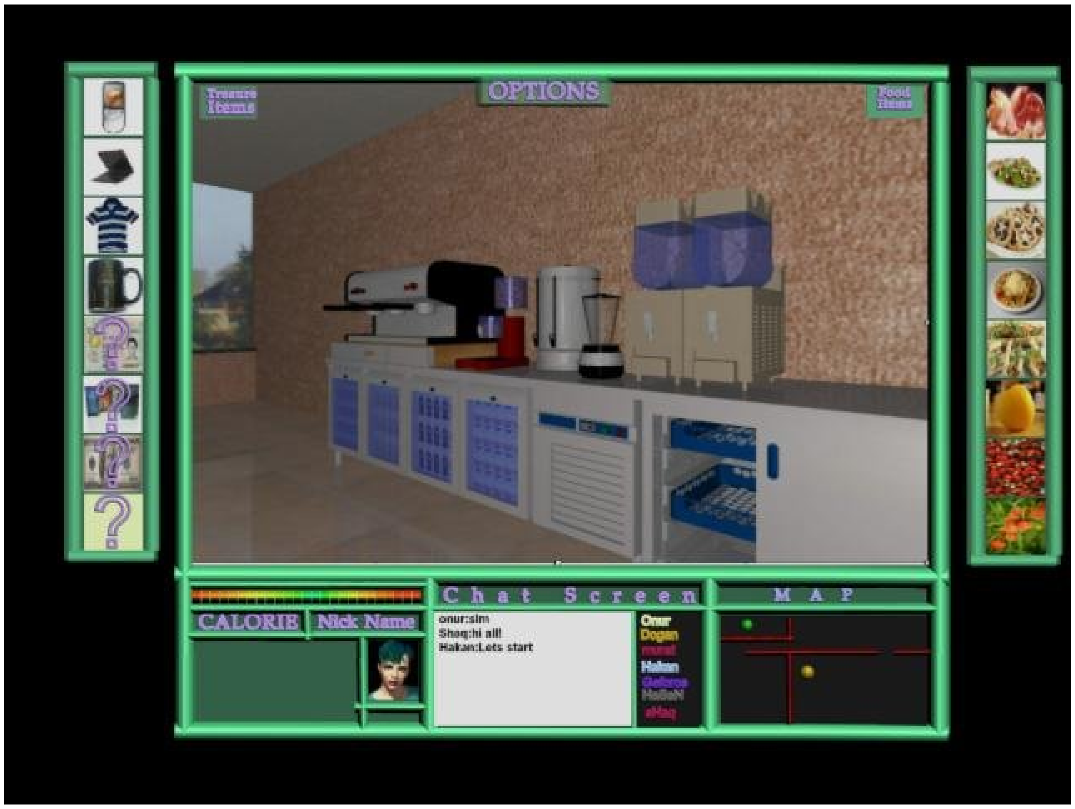


Interface 3.



onur:slm
Shaq:hi all!
Hakan:Lets start

Interface 4.



onur:slm
Shaq:hi all!
Hakan:Lets start

3.2 Game Engine

The game engine will be implemented as a module. It will involve data about game progress and some member functions to uppdate these data accordingly.

3.3 Input Module

We will implement an input module that will handle with all the input data through the use of mouse, keyboard, and console. The mouse, keyboard, and console events will be captured by frame listener binded with OGRE window and necessary callback functions will be called.

3.4 Menu Module

The game will include two menus that provide the user to interact with the game. The content of these menus is determined, of course, according to the general properties of game.

a) Game Main Menu

This menu involves items to start a game. The items in this menu are:

- **Join Game**

  This item is used to register user for the game of next session. Server starts each session of game in a predefined time period. The player that wants to join the game has to wait current session to terminate.
- **Choose Character**

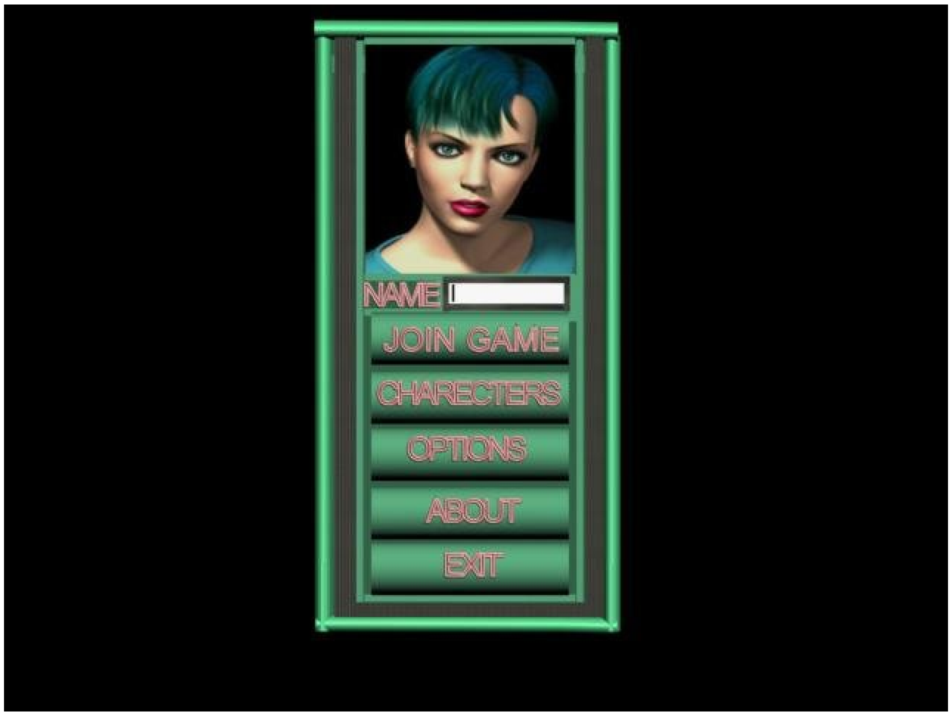
  This item is to make the user select one of pre-defined characters in the game.
- **Options**

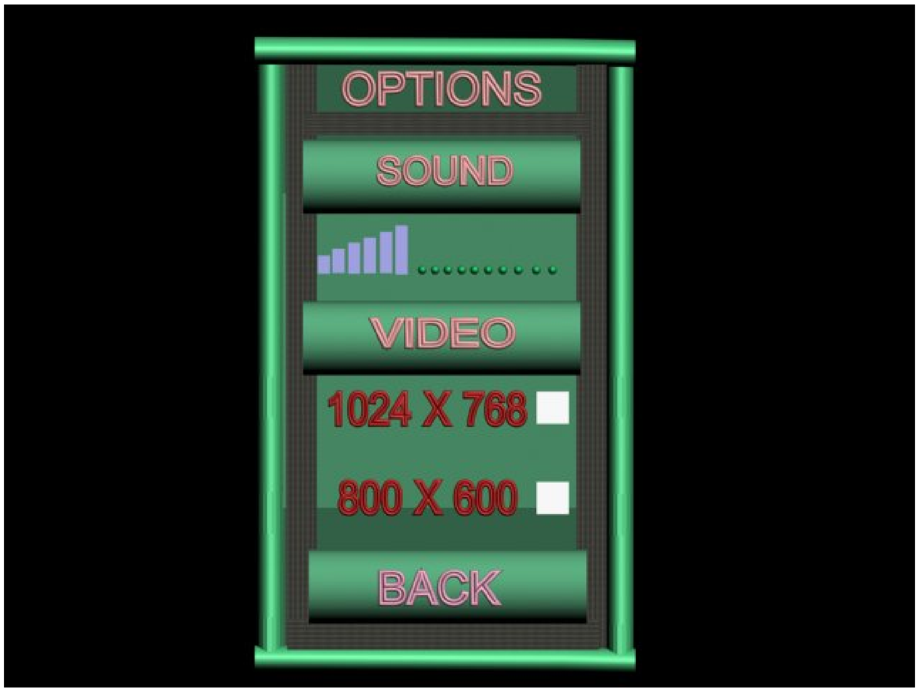
  This item provides change of graphics, audio and keyboard controls.
- **Set Name**

  This item leads user to set name.
- **Exit**

  This item is used to exit from the game.
- **About**

  About us and game.


NAME
JOIN GAME
CHARECTERS
OPTIONS
ABOUT
EXIT


OPTIONS
SOUND
VIDEO
1024 X 768
800 X 600
BACK

b) In Game Menu

- **Objectives**

  The last task assigned to that user is seen by this item.

- **Inventory**

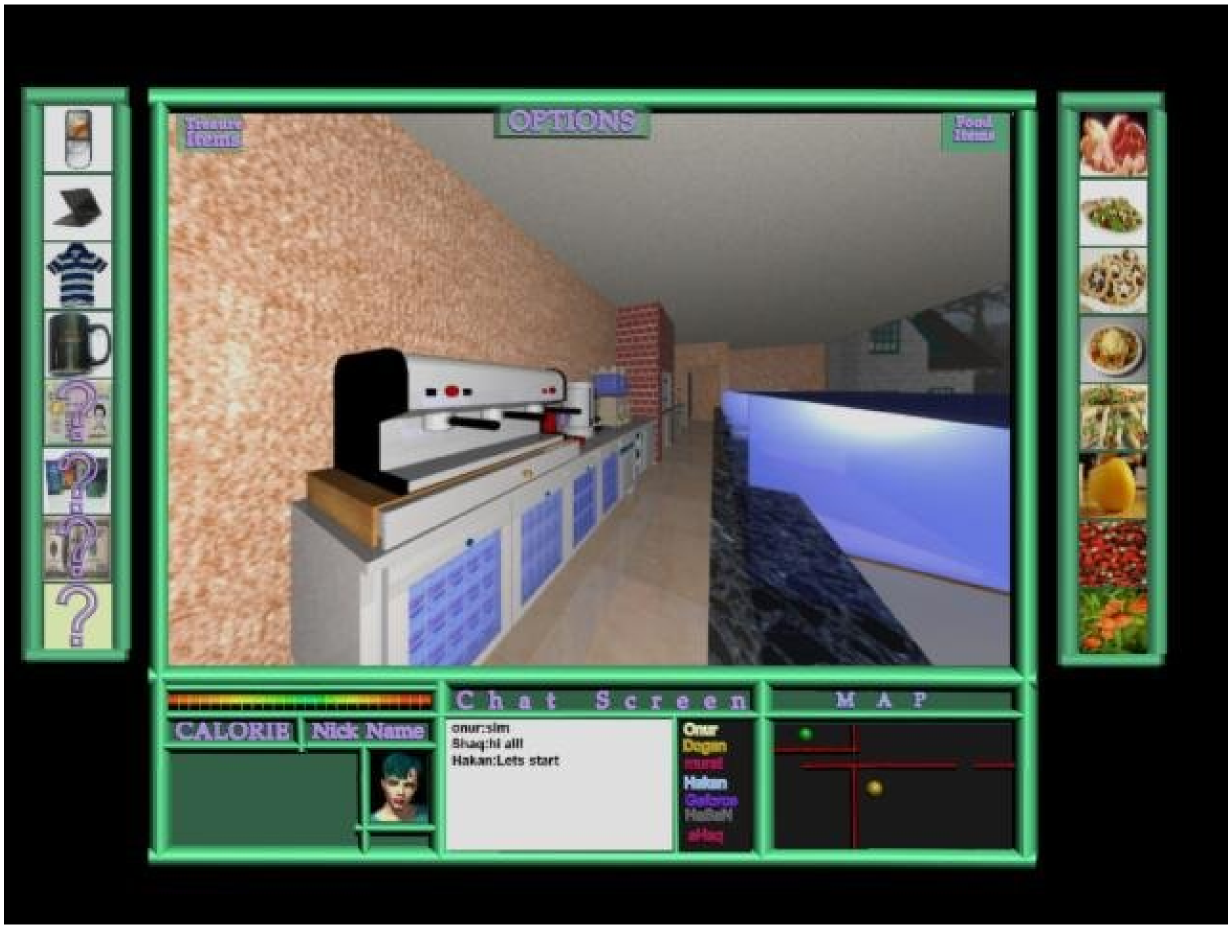
  Inventory that player had get so far.

- **Map**

  A 2D map of the current storey is observed from top view by this menu item.

- **Chat**

  Player can chat with each other.



onur:sim
Shaq:hi all!
Hakan:Lets start

c) Game Pause Menu

By this menu only the player pauses.

- **Leave Game**

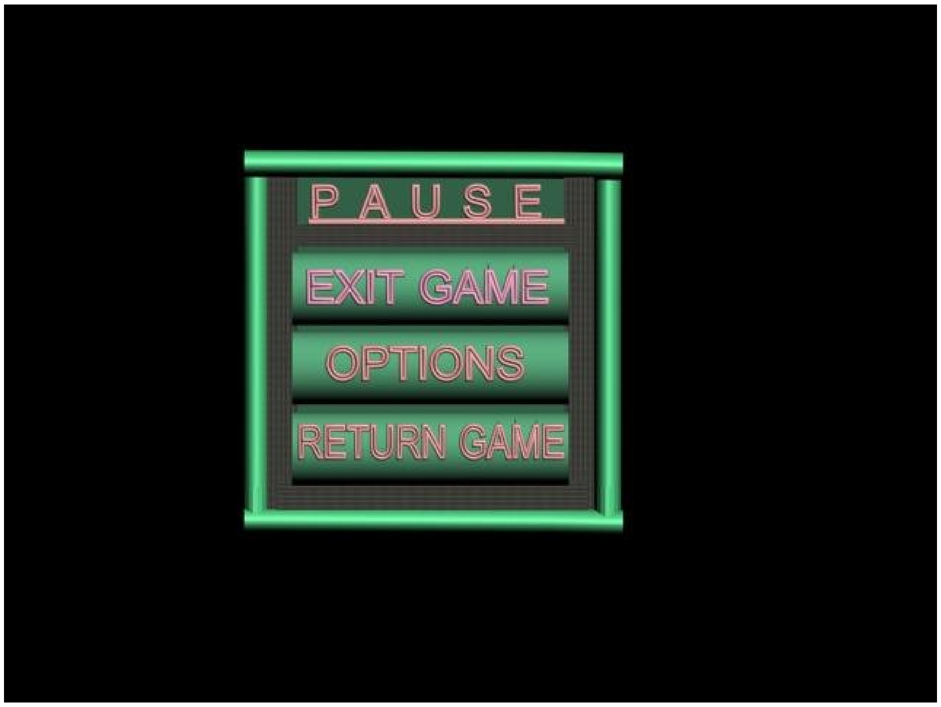
  The user will quit from the current game session.

- **Options**

  This item provides change of graphics, audio.

- **Return To Game**

  This item provides to resume game.



3.5 Artificial Intelligence Engine Module

Being a MMOG, one should be able to play the game although there's no one online besides him/her. This requires implementation of some AI. For this purpose, our game involves Virtual Players (a.k.a. VP). These players ,controlled by an AI engine, has the same attributes and methods as the real players. However, they are not controlled by human players. So that, if an action needs to be taken, action is not triggered (i.e. call to the related [VP prefixed] member function) by a user input, but from the AI engine according to the current situation & progress of the character, and state of the environment. Namely, if the energy level of the character is under some predefined value, then finding food has more priority. If not, then the VP does not spend time and energy to take food, but tries to find the treasure as early as possible (i.e. finding treasure has more priority). And throughout this process, the VP does not go into a loop or does not visit the same room multiple times for the same purpose.

3.6 Graphic Engine Module

Graphic Engine is the one of the most important modules of our project. Graphic Engine will handle all of the rendering operations during the game according to user input from player input and feedbacks from game engine.

We will use OGRE 3D (Object-Oriented Graphics Rendering Engine) as our graphic engine. OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, flexible 3D engine written in C++ designed to make it easier and more intuitive for developers to produce applications utilizing hardware-accelerated 3D graphics. The class library abstracts all the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other intuitive classes. . Because our game project will be a highly object oriented project and integration of OGRE 3D is easiar relatively we decided to use this engine for the rendering issues of the scene.

3.7 Network Module

Treasure hunt is a massively multiplayer online game connecting players through the internet .It is a real time application with the interaction of server and clients.Since it is a real time process, we have to handle some problems because of some external constraints such as connection rate and speed.

Designing the network module and handling problems is one of the most important work for Treasure Hunt.

We will run game engine on server part on the other hand graphic engine, audio engine, pyhsics engine on the client parts.Also we think that we will provide a second computer as a server to run the ai engine.This will be done for efficiency constraints.Server will have a database schema keeping the position of each character and the step the player is in.

Server will send the position of other players to a player if they are in the same room.A time period of 50 ms will be enough.We have estimated that if we send the position in every 50 ms accuretly there will be no lag in the game.And if a player takes a treasure or food object the object_id of this object will send to server.And server will send the object_id immediately to the remaining clients as the object is now passive.

We will have one server and many clients.Server will be the fastest computer with the best connection and other computers will be clients. Although there are many ways of encoding packets, they are all transmitted as either UDP or TCP packets.TCP packets are very good for transferring data, but TCP packets are not so good for games.Because of the TCP protocol, TCP packages are often delayed (resulting in games with a lot of lag) and arrive as streams rather than packets.As a result we have to implement our scheme to separate the data.On the other hand UDP packets are very good because they are sent right away and sent in packets. Therefore data can easily be distinguished.Since efficiency is another constraint it is also good to use UDP protocol.Since TCP has a handshake protocol.

However UDP have some disadvantages and we have to deal with below problems :

UDP packets are not guaranteed to arrive.All the packets that we send, some fraction or possibly none of the packets could be get.For example a player could pick the treasure but this information would be lost.As a result, player could not see the treasure object of the next step, altough he has picked the object of previous step,.

UDP packets are not guaranteed to arrive with the same order it has been sent.It would be a huge problem for our game.Because it is an important concept in our game who does before.

UDP packets have no protection from hackers but it is not a important task to handle in this step.

UDP transport does not provide flow control or aggregation so it is possible to overrun the recipient and to send data inefficiently.

Another problem according to the connection rate and speed is explained in this scenario: A player gets a food object and the object_id will be send to server.After that the server will send all the clients that the food has taken.So food object will dissappear.But what can be done if another player gets the same food with same object_id during this period (client to server and server to cients).

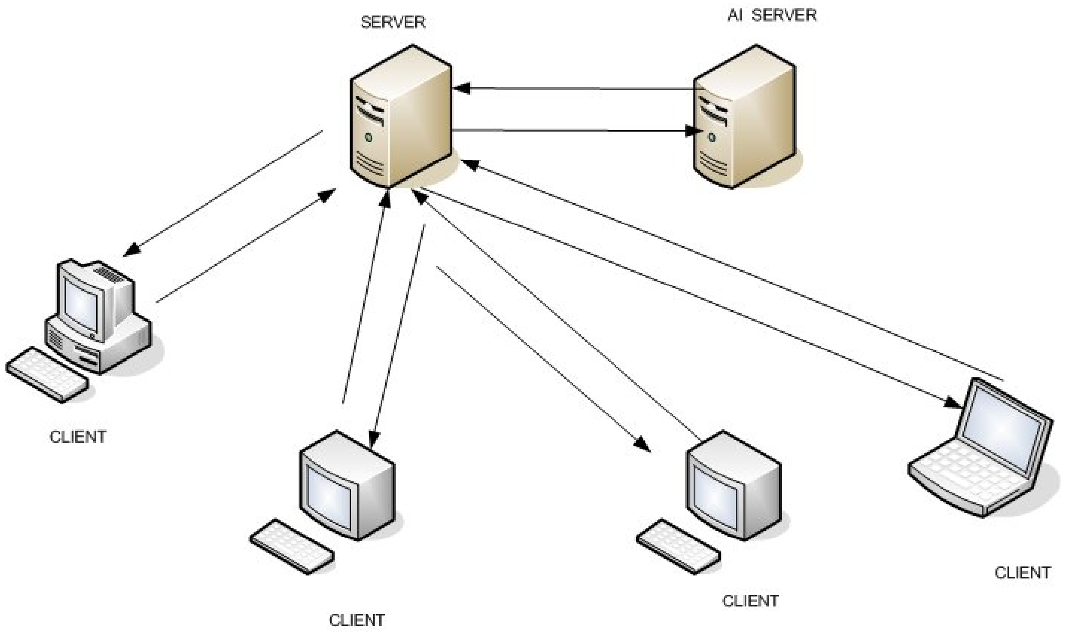
Morever another problem is that if the second players' information comes before the first.

We have to define an error management system. Assuming the internet is not reliable,we have to handle connection problems rather than block, lock-up, or crash.

We have to define a resend package method for the packages that does not arrive.

We have to define order and sequence method that the packages arrive out of order.

We have to define a flow control and aggreagation method.



3.8 Audio Module

Game will have different soundtracks during game and waiting for the next session. The soundtrack will change between the steps. Sound effects will be used during games.
Sound options will be controlled and volume level is adjusted by user.

We will use FMOD as our sound library. Fmod provide us all our needs. The most important feature of FMOD sound library for us is that using minimal resources and being scalable. We will use sound library in two place in our game. One for environment sounds like footsteps. And the other one for playing game musics. FMOD also provide us to play multiple sounds simultaneously, so that we can play some game music and environment sounds at the same time.

The Audio Module will hold the path of the selected sound file and load the file to memory when desired. The Audio Module will call the required functions from the fmod library. Player can input the properites of audio module like play/stop music, change volume of music from the user interface. After initialization in the beginning of the program, it starts playing when program call ‘play’ and stops when we call ‘stop’ function.

3.9 Chat Module

Chat module will enable players conversing with other players,in real time.Players will be able to post his/her message in a chat window and watch others post their messages on the game screen. During game interaction between players make some lively dialogue,and a much more enjoyable game. Moreover user can send some sound effects.When they find a treasure object he can send a sound effect. A sound effect can be a laugh,haw-haw,"you have no chance","go your home","I am the hunter","Yeah I have found".

4. CLASS DEFINITIONS

4.1 BaseObject Class

- **Attributes:**

It is the base class that treasure object, food object and furniture object that inherit from.

object_name: It is a unique string value to each object which identify the object.We will have treasure object, food object and furniture object and all of them will have a unique name such as key, book, laptop or cake,apple,bread or table,chair and computer.

object_id: Objects which have the same name will uniquely identify which object it is.

object_position: It is a pointer to position object.

angular_direction: It is a float number that provides the direction of object according to the angular constraints.
model: the name of model object that will be used while importing objects.(such as model.3ds)

texture: the name of texture file that will be used while importing object.

- **Methods:**

void init_model(): Initializes all Object3DS instances with its respective *.3DS file from an external resource using void initialize() method. It also initializes position and scaling factor of object using void set_scalar() and void set_position() method.

void initialize(): Initializes Object3DS instances from a source of *.3DS file using model and texture attribute.

void set_position(): Sets the position of objects according to the angular direction using rotate and translate methods.

void render(): The top-level drawing function. This function makes all calls necessary to create the world.

void disappear(): This function makes picked food and treasure objects invisible. Also if the player is at the first step of treasure hunt he/she cannot see the treasure objects related to remaining steps.

void appear(): This function makes treasure objects visible. If the player passes the first step of treasure hunt he/she can see the treasure objects related to next step.

void set_scalar(): This funcion is used to scale the model from its natural size.

void translate(): This function will be used to translate 3D objects.

void rotate(): This function will be used to rotate 3D objects.

4.2 Position Class

- **Attributes:**

x_coordinate : It is a float value indicating the x coordinate of the object in xy plane.

y_coordinate : It is a float value indicating the y coordinate of the object in xy plane.

z_coordinate : It is a float value indicating the relative z coordinate of contact point of the object.

floor_info : It is an integer value indicating at which floor the object is.

- **Methods**

  Get_Position() : gets position attributes.

  Change_Position() : changes position attributes.

4.3 Treasure Object Class

- **Attributes:**

  Object_name, object_id,object_position,angular_direction,model and texture attribute will be inherited from base object class.

  is_active=If the player takes the treasure object the object will lose brightness (inactivated) so other players cannot take this object.

  Books, Keys, Phone, Notebook, Cup, Money, Pass Card, T-shirt, Pants, Shirt, Socks, Tea-pot, Life Preserver, Tape Recorder.

- **Methods:**

  void init_model():

  void initialize():

  void set_position():

  void render():

  void disappear():

  void appear():

  void set_scalar():

  void translate():

void rotate():

Methods above will be used from the base class by inheriting.

void activate_deactivate(): If the player takes a step related item it will be deactivated for all other but the next steps' item will be all activated.

4.4 Food Object Class

- **Attributes:**

is_active: If the player pickes the food object the object will lose brightness (inactivated) so other players cannot take this object.

calorie: It is the integer value that shows the calorie value of the food object.

is_animal: It is the boolean value that shows the food is pertaining to an animal. Vegetarian players could not eat this type of food.

Cake: calorie =200 is_animal=0	**Coke:** Calorie=80 Is_animal=0
Bread: Calorie=100 Is_animal=0	**Ice-Cream:** Calorie=100 Is_animal=0
Cheese: Calorie=50 Is_animal=0	**Hamburger:** Calorie=100 Is_animal=1
Banana: Calorie=50 Is_animal=0	**Fish:** Calorie=200 Is_animal=1

Chocolate: Calorie=150 Is_animal=0	**Buttock:** Calorie=300 Is_animal=1
Watermelon: Calorie=30 Is_animal=0	**Chicken:** Calorie=200 Is_animal=1
Apple: Calorie=20 Is_animal=0	

- **Methods:**

void init_model():

void initialize()

void set_position():

void render():

void disappear():

void appear():

void set_scalar():

void translate():

void rotate():

Above methods will be used from the base class by inheriting.

void deactivate(): If the player takes a food item it will be deactivated for all others.

void eat(): If the character does not have any allergic problem to the founded food object the calorie of the food will be added to energy of character.If it has it would not add.

4.5 Furniture Object Class

Table, Chair, Wardrobe, Personal Computer, Refrigerator, Sofa, Blackboard, Flower, Garbage.

- **Methods:**

  void init_model():

  void initialize():

  void set_position():

  void render():

  void disappear():

  void appear():

  void set_scalar():

  void translate():

  void rotate():

  These methods will be used from the base class by inheriting.

4.6 Character Class

- **Attributes**

  name : It is a string unique to each character which is given by player or if it is virtual player this name is given by game engine.

  type : It is an integer to indicate type of character, which are predefined by us.

  status : It is an integer indicating the status of the character as ‘Stand’, ‘Slow Walk’,‘Walk’, ‘Run’

energy : It is an integer to determine the speed of character. Throughout the game, this value will decrease from its initial value, and can be increased by eating food found in rooms. If this value is under a threshold value, character can only do ‘Slow Walk’.

position : It is a pointer to Position object.

crouch: It is an integer indicating whether the character is crouching or not. So that in raycasting the camera position is lowered.

inventory : it is an array of object_names that have been taken.

direction: Its is float to determine the angle between character's current direction and x axis.

allergic_foods : it is array of object_names that cannot be eaten by character.

is_vegetarian : it is an integer indicating whether the character is vegetarian or not. If so animal foods can not be eaten by character.

is_virtual : It is an integer indicating whether the character is a virtual player or not. If so, VP_Base_Function is called throughout the game to control the virtual player.

- **Methods:**

Move(): This function gets an input from player and according to that input changes the position and status of character, if possible (i.e. There is not any object in front of the character). To change the position it uses some information of the character. To determine the direction of move it uses the **direction** of the character. The current **energy** of character and input from player is used to determine the speed.

Rotate(): This function gets an input from player and according to that input changes the direction of character.

Pick_Treasure() : This function provides taking a treasure object. If called, this function removes the object from the map and add that object to the inventory of the player**.**

Pick_Food() : This function provides taking a food object. If called and the food is not in allergic_foods, this function removes the food object from the map, and increases the energy of the player according to energy value of that food**.**

Decrease_Energy() : This function is called in main loop according to time value of game. This function is called every 10 seconds, and it decreases the **energy** by an amount of 30 calories.

4.7 Virtual Player Class

This is a class inherited from Character Class.

- **Attributes:-**

The ones inherited from character class.

- **Methods**

VP_Base_Function() : This is the function to control the virtual player(VP) throughout game, and according to the situation & energy of the VP, it decides to call helper functions (either VP_Find_Food(), or VP_Find_Treasure()).

VP_Find_Food() : This function is called by VP_Base_Function() when the energy level is under a predefined threshold level. When called, required Move(), and Pick_Food() functions are called by this function till finding some food to gain energy.

VP_Find_Treasure() : This function is called by VP_Base_Function() when the energy level is over a predefined threshold level. When called, required Move(), and Pick_Food() functions are called by this function till VP finds the trasure object or the energy level of VP falls under the treshold.

4.8 Map Classes

4.8.1. POINT

- **Attributes**

position : It is a pointer to **POSITION** object.

- **Methods**

Get_Position() : get position of point.

4.8.2 WINDOW

- **Attributes**

window_id : It is an interger unique to each window.

first_point : It is a pointer to **POINT** object.

second_point : It is a pointer to **POINT** object.

width : it is a float indicating width of window.

heigth : it is a float indicating heigth of window.

heigth_from_ground : it is a float indicating heigth of window from ground.

- **Methods**

Draw_Window() : draw the window

4.8.3 DOOR

- **Attributes**

door_id : It is an interger unique to each door.

first_point : It is a pointer to **POINT** object.

second_point : It is a pointer to **POINT** object.

width : it is a float indicating width of window.

heigth : it is a float indicating heigth of window.

is_open : it is a bool indicating whether open or not.

- **Methods**

Draw_Door() : draw the window.

Open_Door() : open the door.

Is_It_Open() : is this door open or not

4.8.4 ROOM

- **Attributes**

room_id : It is an interger unique to each room.

corners : It is an array of room floor polygon's corner **POINT**s.

doors : It is an array of **DOOR**s of room.

windows : It is an array of **WINDOW**s of room.

objects : It is an array of **OBJECT**s of room.

- **Methods:**

  Draw_Room() : Draw this door.

4.9 Network Class

Server will have a database schema keeping the position of each character and the step the player is in.

- **Attributes**

  player_id: it is an integer which is unique to all user.

  ip_no: it is an integer number which keeps the related ip number of all other objects.

  Position *object_postion;

  character_position : It is a pointer to position object.It takes the position of a character.

  taken_tre_object_id[]: It is an integer array of object_ids of the taken treasure objects.

  taken_food_object_id[]: It is an integer array of object_ids of the taken food objects.

```
struct serverdatabase{
player_id
taken_tre_object_id[]=
ip_no
character_position
}
```

- **Methods (server)**

  send_data(): Sends the related data to clients.It has a resend method package for the data doe not arrive to clients.

listen_port(): Listens the same port for all clients.

take_data(): Takes and handles the data sent by client.

Server will send the position of other players to a player if they are in the same room.And if a player takes a treasure or food object the object_id of this object will send to server.And server will send the object_id to the remaining clients to tell the objects are now passive.

- **Methods (client)**

send_data(): Sends the related data to clients.It has a resend method package for the data doe not arrive to server.

take_data(): Takes and handles the data sent by server.

order_packages(): Orders that the packages arrive out of order

4.10 Audio Class

- **Attributes**

audio_id : it is unique integer indicating the id of audio data.

volume : it is float indicating volume of sound.

- **Methods:**

Play_Audio() : This function play the audio which's id is **audio_id.**

Stop_ Audio() : This function stop the audio which's id is **audio_id.**

Set_Volume() : This function set the **volume** of audio**.**

Get_Volume() : This function get the **volume** of audio**.**

Get_Audio_File() : This function get the audio which's id is **audio_id.**

4.11 GameEngine Class

This is a class handling the game progress.

- **Attributes:**

  position *current_position: this is a pointer to a position type object, holding the current position of the character.

  int objective: holds the object_id of the object, that is assigned to character to find.

- **Methods:**

  Update_Environment_Info(): When called, this function updates the environment information.

  Update_Objective(): This function is called when an objective is finished (i.e. assigned object is found). If the objective is not the last one(i.e. the found object is not the treasure), the **objective** attribute is updated according to the new objective.

4.12 Input Class

- **Attributes:**

  chat_message : it is a string value that the player writes on the chat console.

  mouse_input : it is a position object that specify the place that mouse clicked

  keyboard_input : it is a key from keyboard that have a functionality in our game. Most probably the keys that have functionality in our game will be Enter, Up, Down, Left, Right, Space and etc.

- **Methods**

  void Get_Message_From_Mouse() : Get mouse input

  void Get_Message_From_Keyboard(): Get keyboard input

  void Get_Message_From_Console(): Get the message written to the chat console

  void Send_Message_From_Mouse() : Send mouse input to the appropriate classes

  void Send_Message_From_Keyboard(): Send keyboard input to the appropriate classes

  void Send_Message_From_Console(): Send the message written to the chat console

4.13 AIEngine Class

VIRTUAL PLAYER

This is a class inherited from Character Class.

Attributes: -

Methods:

VP_Base_Function() : This is the function to control the virtual player(VP) throughout game, and according to the situation & **energy** of the VP, it decides to call helper functions (either **VP_Find_Food()**, or **VP_Find_Treasure()**).

VP_Find_Food() : This function is called by **VP_Base_Function()** when the energy level is under a predefined threshold level. When called, required **Move()**, and **Pick_Food()** functions are called by this function till finding some food to gain energy.

VP_Find_Treasure() : This function is called by **VP_Base_Function()** when the energy level is over a predefined threshold level. When called, required **Move()**, and **Pick_Food()** functions are called by this function till VP finds the trasure object or the energy level of VP falls under the treshold.

4.14 GraphicsEngine Class

- **Attributes**: -
- **Methods:**

void Draw_map(): It will render the parts of the building that are visible to the player.

void Draw_furnitures(): It will render the the furnitures visible to the player.

void Draw_objects(): It will render the objects looked for by the player.

void Draw_treasure(): It will render the Treasure, if it is visible by the player.

void Draw_characters(): It will render the charecters visible to the player.

void Draw_food(): It will render the foods visible to the player.

void Draw_All(): It will call the methods above in the correct order.

4.15 Chat Class

- **Attributes:**

  static Audio *audios; This audio will contain the audio files played when the user send a sound effect.

  static int sound_id;This integer will be unique to audio files

- **Methods (server)**

  send_data() : This function will be inherited from network class.

  take_data() : This function will be inherited from network class.

  send_sound_effect(): This function will send the sound_id of the object to all clients.

- **Methods (client)**

  send_data() : This function will be inherited from network class.

  take_data(): This function will be inherited from network class.

  take_sound_effect(): This function will send the sound_id of the object to all clients.

  play_sound_effect(): This function will play the audio of the related sound_id.

5. TOOLS

Throughout the implementation of our game, needs will arise by means of different aspects. After some research and analyze we decided on following issues.

- Our game will run on **Microsoft Windows XP** operating system. The reason is its wide usage and compatibility with other tools which will be used throughout the project.

- We will develop our game on **Microsoft .NET** platform.

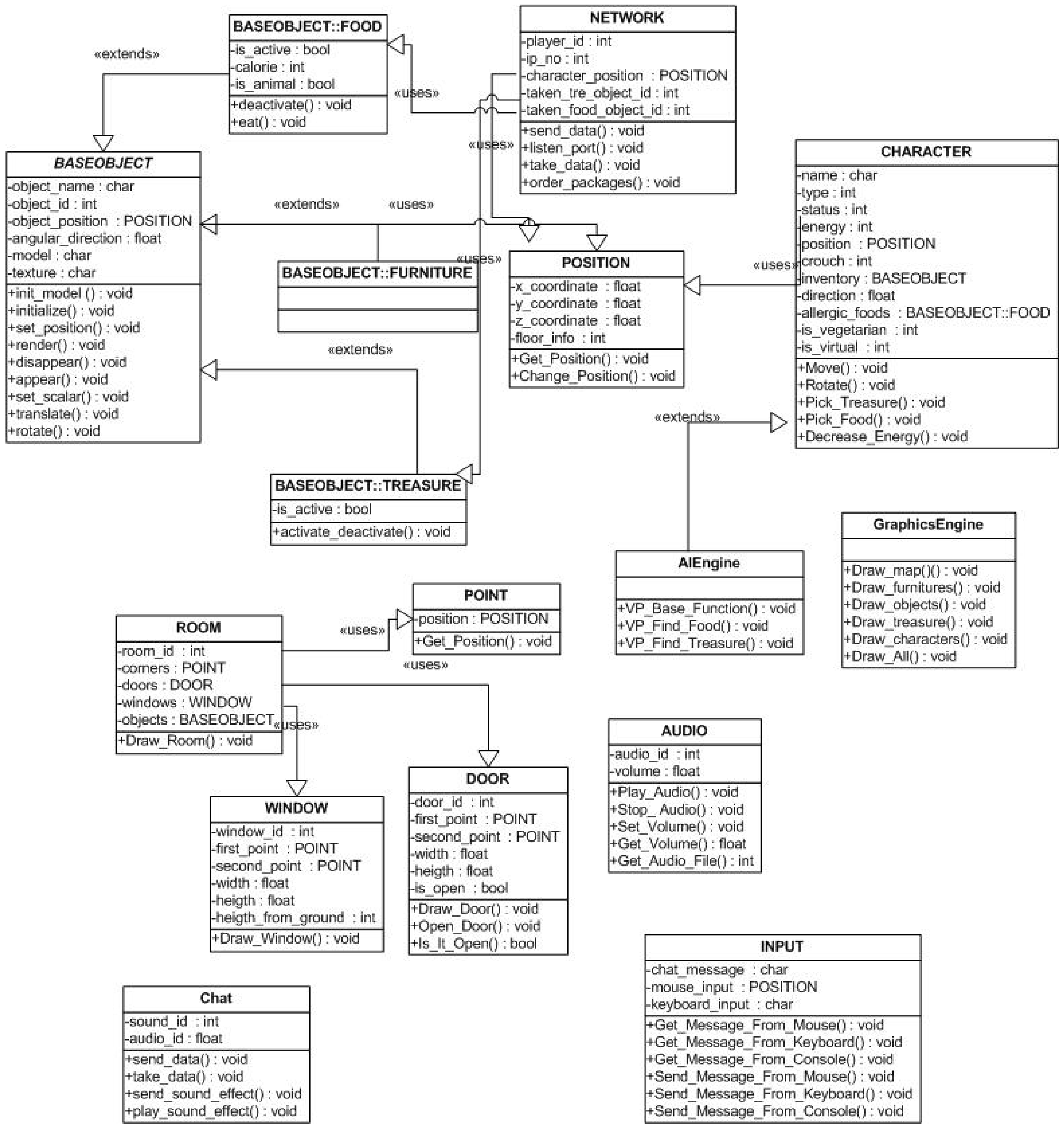
- In terms of graphics, we will use **C++** for implementation and **OpenGL** as graphics library.The reason is being experienced about C++ programming language and OpenGL as a team.

- We will also use **OGRE** (Object-Oriented Graphics Rendering Engine). Being a flexible open-source engine and implemented using C++, makes it a considerably helpful tool for our project.

- Handling images and textures raises a need to use a tool. For this purpose, we will be using **Adobe PhotoShop**, being one of the most poweful & easy-to-use tools, and experience of team members.

- Another need arises by means of handling 3D modelling issue. **3D Studio Max** will be used for this purpose

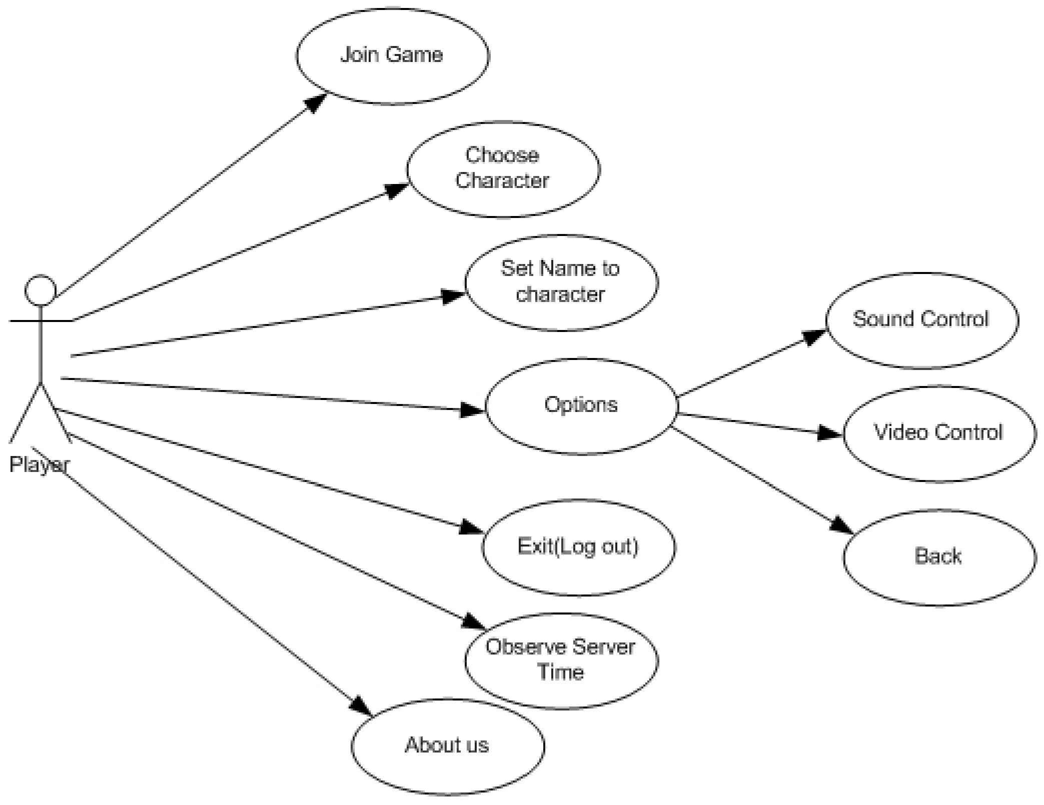
6. CLASS DIAGRAM



7. USE CASE DIAGRAM

a)Game Main Menu Use Case:

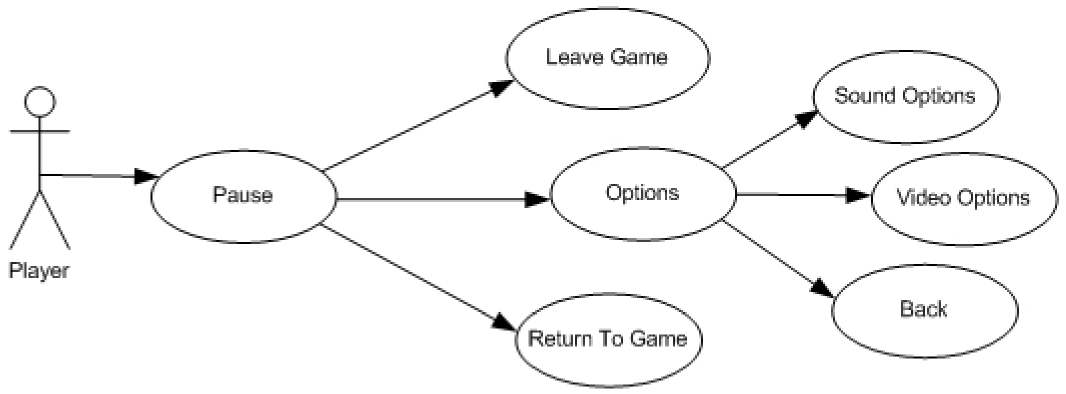
These are the use cases that user will face while joining the game.



GAME MENU

b) Game Pause Menu

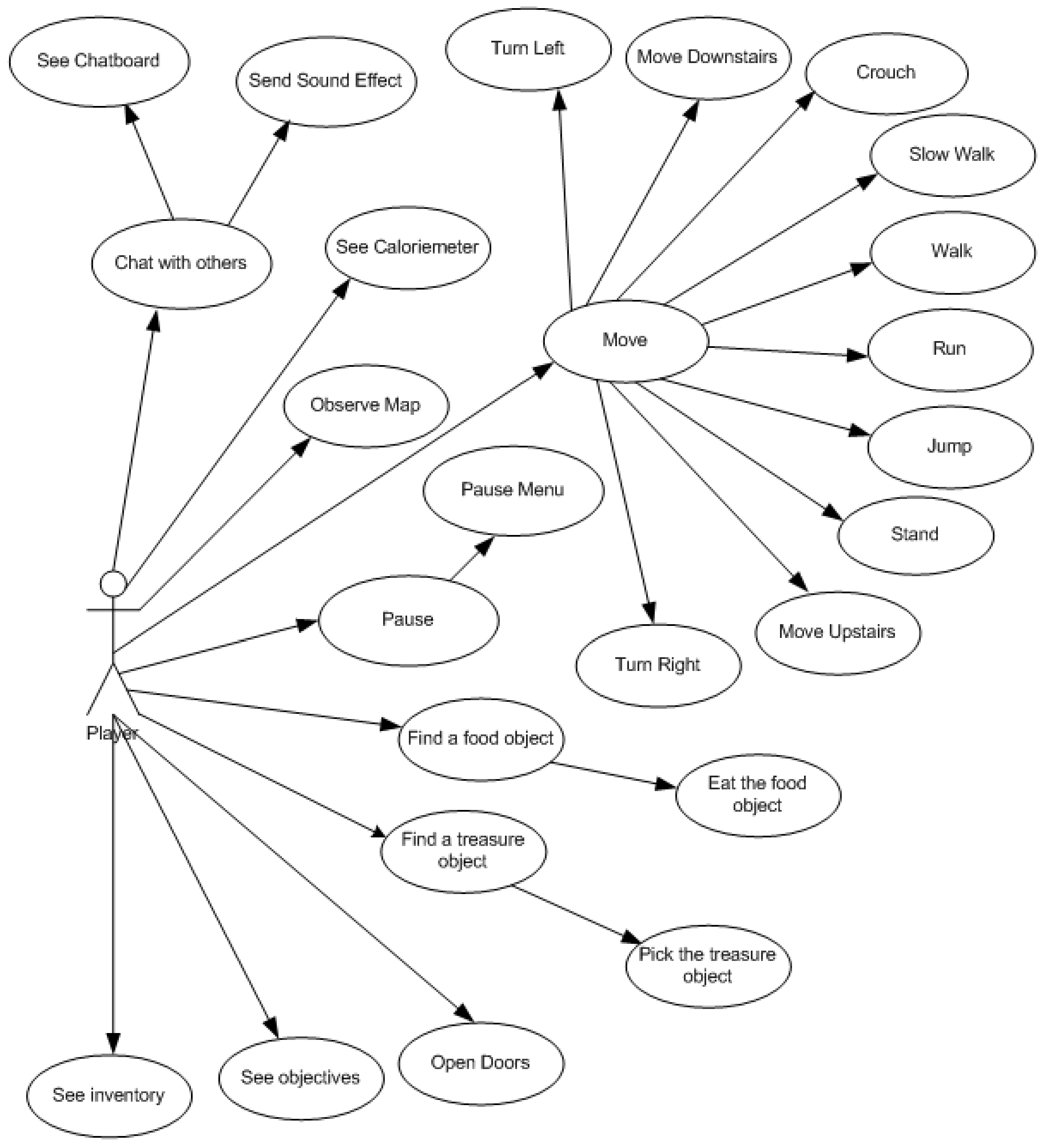
These are the cases that user can do while pausing the game.



PAUSE MENU

c) In Game Menu Use Cases

These are the use cases that user will face while playing the game.



IN GAME MENU

8. DFD

8.1 Data Flow Diagram Level 0

Level 0 DFD in Figure 1 of 9.1 is the overall system. It shows the interaction between user (game client) and game server, it also shows the interaction beetween game AI machine and game server .

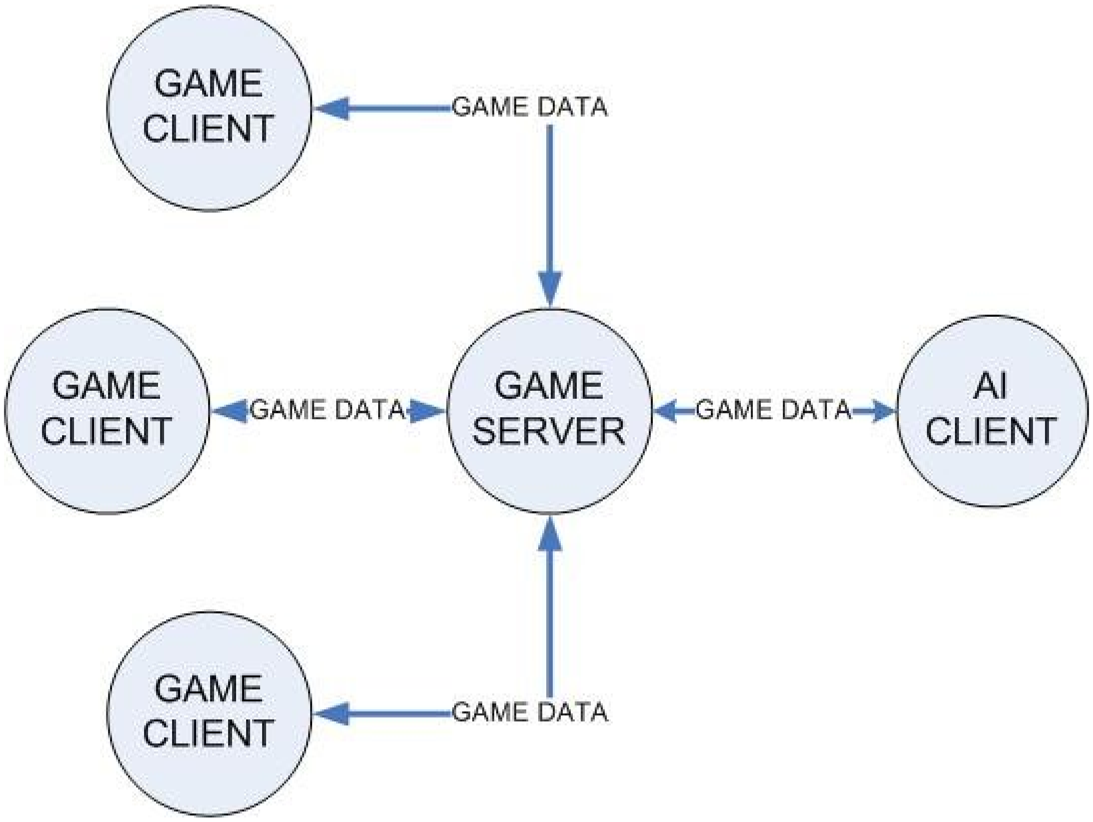


Figure 1 of 8.1

8.2 Data Flow Diagram Level 1

Game Client

Figure 1 of 8.2 is more detailed look on overall system in game client. Innerstructures and sub-modules are described in a detailed way. Player will provide inputs using keyboard and mouse and take graphics and audio outputs. Game client sends scene data to graphics engine and graphic engines sends graphics outputs to the game client. Game client sends game state to audio and player take audio input.

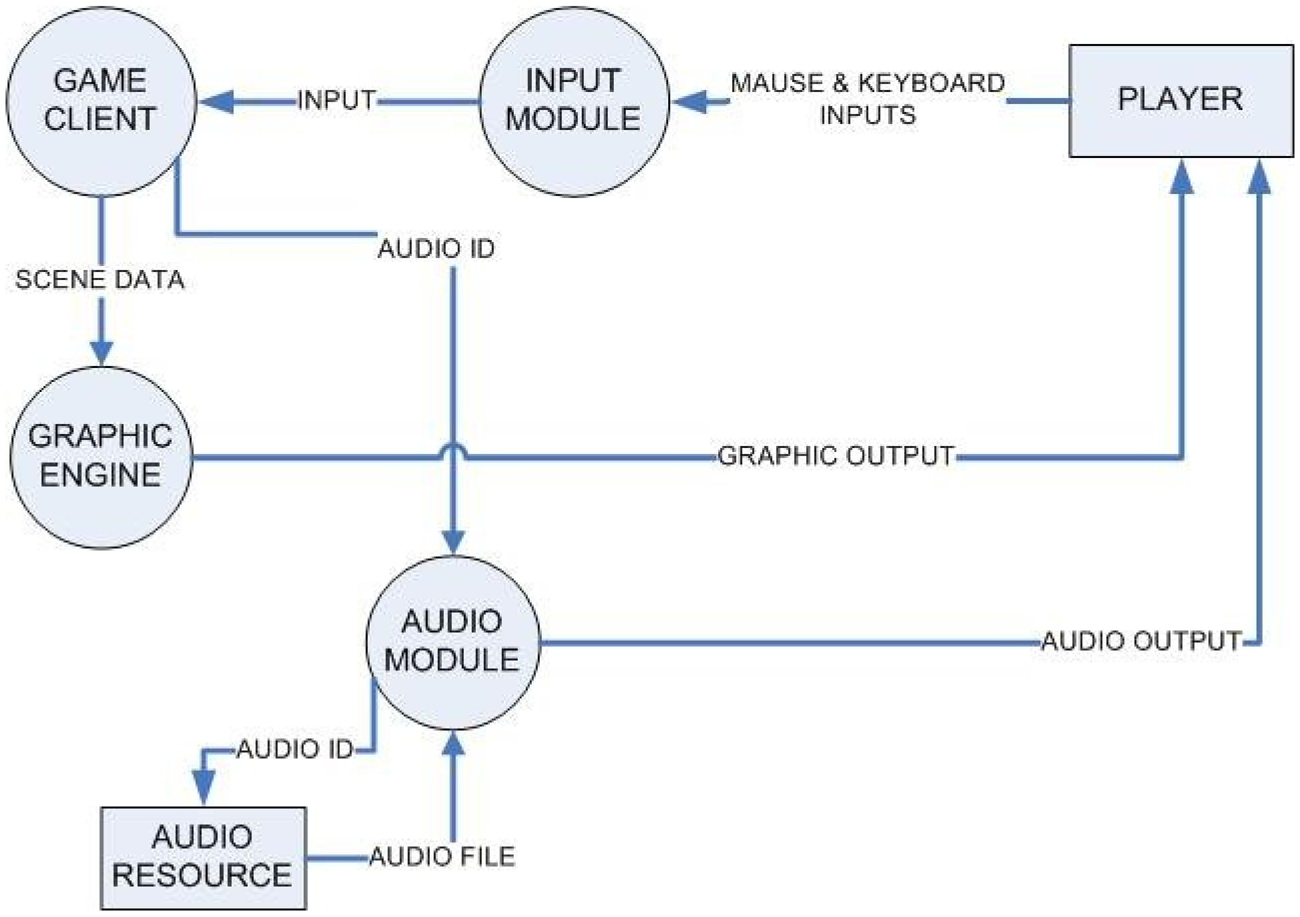


Figure 1 of 8.2

Game Server

Network component provides data transfer between game client and game server in Figure 2 of 8.2. Game server sends game data to AI cleint and takes AI players (virtual characters) data, such as its behaviour, its coordinates.

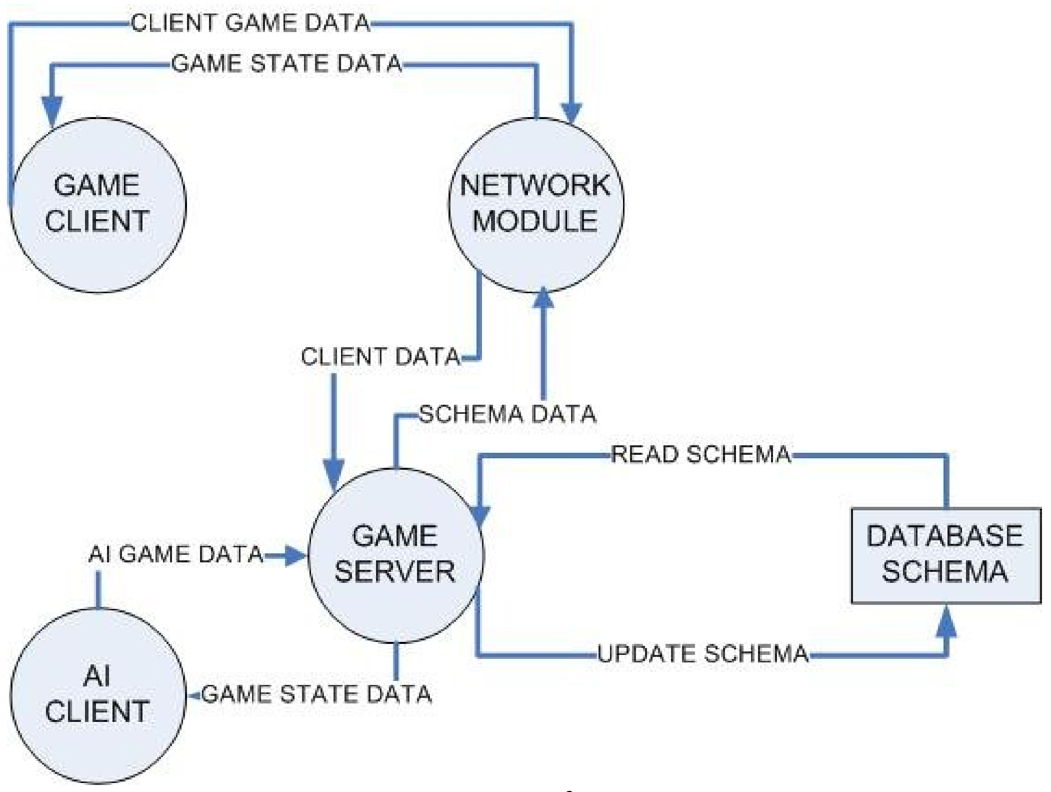


Figure 2 of 8.2

8.3 Data Flow Diagram Level 2

Network Module

Game Data is transferred from clients to game server, game server to clients by network engine and with use of a database schema in Figure 1 of 8.3

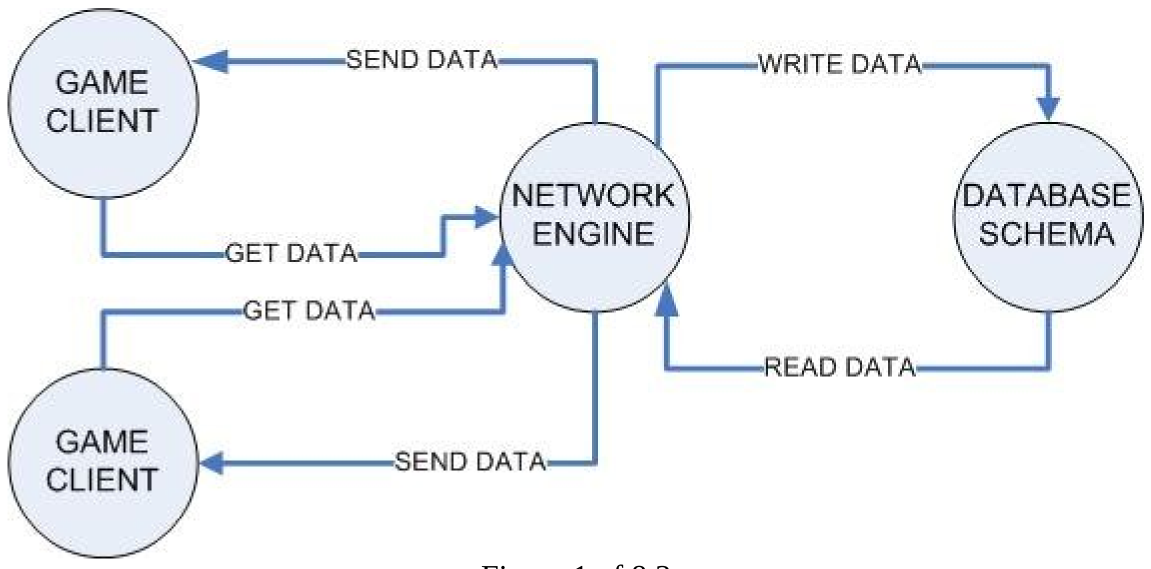


Figure 1 of 8.3

Input Module

Input module that will handle with all the input data through the use of mouse, keyboard to graphic and chat angine in Figure 2 of 8.3

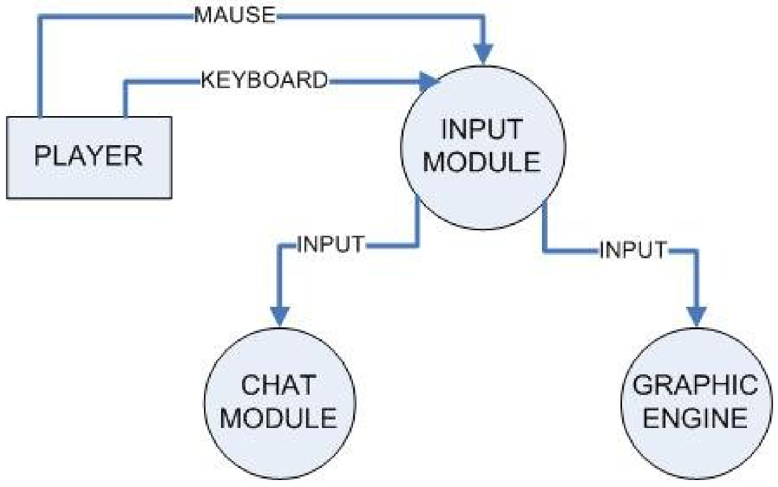


Figure 2 of 8.3

Audio Module

The Audio Module will hold the path of the selected sound file and load the file to memory when desired in Figure 3 of 8.3

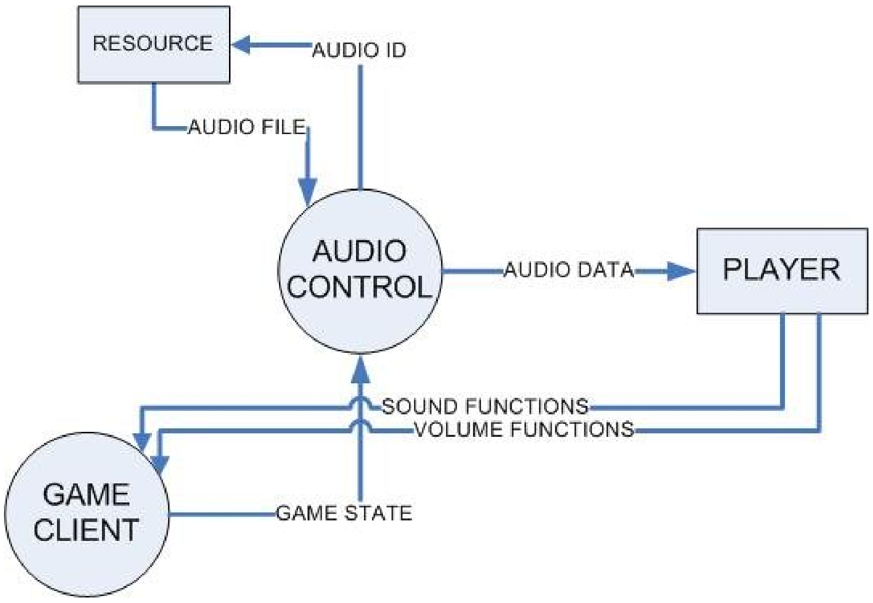
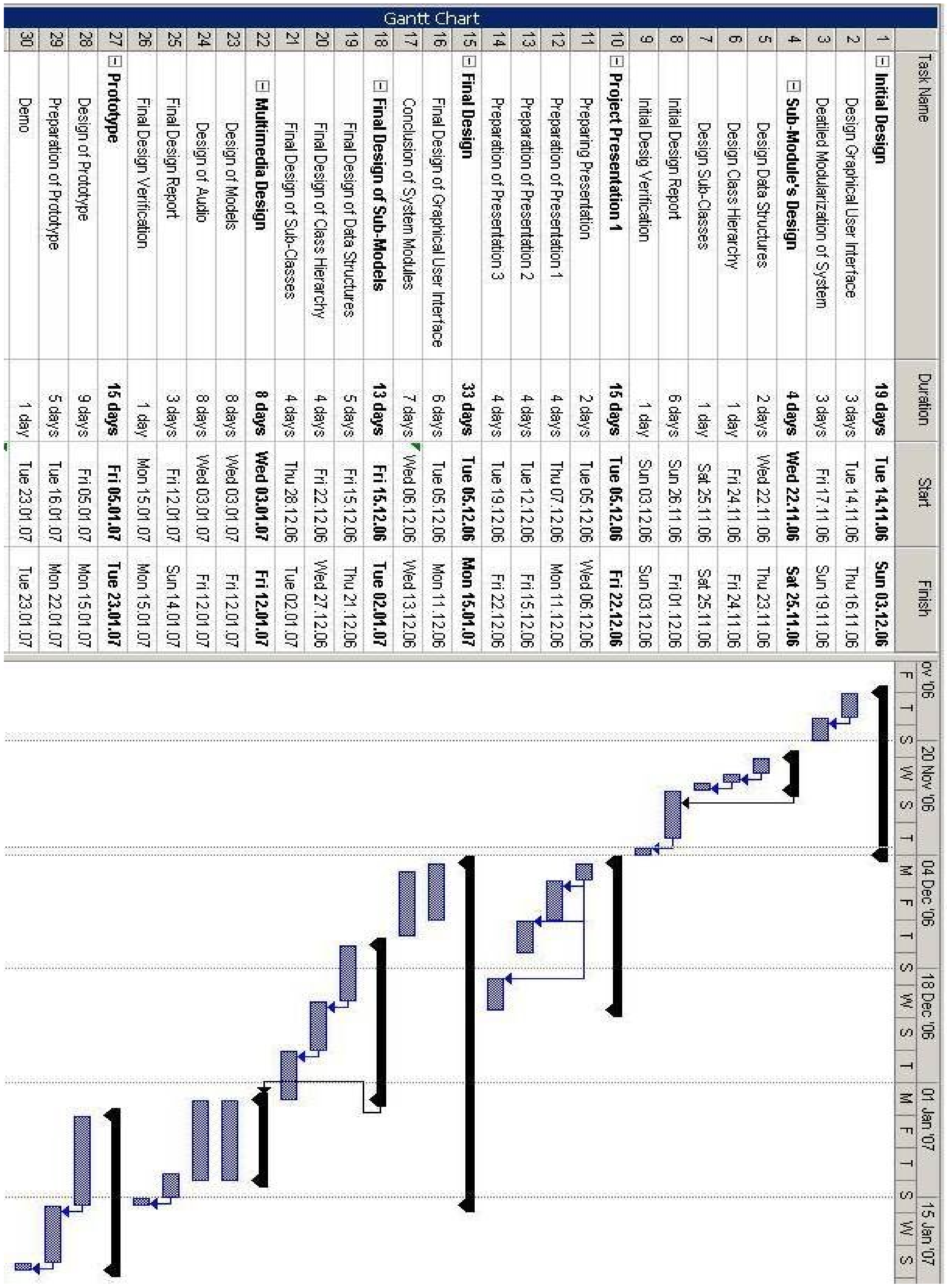


Figure 3 of 8.3

9. Gantt Chart



	Task Name	Duration	Start	Finish
1	**Initial Design**	**19 days**	**Tue 14.11.06**	**Sun 03.12.06**
2	Design Graphical User Interface	3 days	Tue 14.11.06	Thu 16.11.06
3	Deatiled Modularization of System	3 days	Fri 17.11.06	Sun 19.11.06
4	**Sub-Module's Design**	**4 days**	**Wed 22.11.06**	**Sat 25.11.06**
5	Design Data Structures	2 days	Wed 22.11.06	Thu 23.11.06
6	Design Class Hierarchy	1 day	Fri 24.11.06	Fri 24.11.06
7	Design Sub-Classes	1 day	Sat 25.11.06	Sat 25.11.06
8	Initial Design Report	6 days	Sun 26.11.06	Fri 01.12.06
9	Initial Desig Verification	1 day	Sun 03.12.06	Sun 03.12.06
10	**Project Presentation 1**	**15 days**	**Tue 05.12.06**	**Fri 22.12.06**
11	Preparing Presentation	2 days	Tue 05.12.06	Wed 06.12.06
12	Preparation of Presentation 1	4 days	Thu 07.12.06	Mon 11.12.06
13	Preparation of Presentation 2	4 days	Tue 12.12.06	Fri 15.12.06
14	Preparation of Presentation 3	4 days	Tue 19.12.06	Fri 22.12.06
15	**Final Design**	**33 days**	**Tue 05.12.06**	**Mon 15.01.07**
16	Final Design of Graphical User Interface	6 days	Tue 05.12.06	Mon 11.12.06
17	Conclusion of System Modules	7 days	Wed 06.12.06	Wed 13.12.06
18	**Final Design of Sub-Models**	**13 days**	**Fri 15.12.06**	**Tue 02.01.07**
19	Final Design of Data Structures	5 days	Fri 15.12.06	Thu 21.12.06
20	Final Design of Class Hierarchy	4 days	Fri 22.12.06	Wed 27.12.06
21	Final Design of Sub-Classes	4 days	Thu 28.12.06	Tue 02.01.07
22	**Multimedia Design**	**8 days**	**Wed 03.01.07**	**Fri 12.01.07**
23	Design of Models	8 days	Wed 03.01.07	Fri 12.01.07
24	Design of Audio	8 days	Wed 03.01.07	Fri 12.01.07
25	Final Design Report	3 days	Fri 12.01.07	Sun 14.01.07
26	Final Design Verification	1 day	Mon 15.01.07	Mon 15.01.07
27	**Prototype**	**15 days**	**Fri 05.01.07**	**Tue 23.01.07**
28	Design of Prototype	9 days	Fri 05.01.07	Mon 15.01.07
29	Preparation of Prototype	5 days	Tue 16.01.07	Mon 22.01.07
30	Demo	1 day	Tue 23.01.07	Tue 23.01.07